A GUIDE TO CREATING RLE GRAPHICS ON THE T/S 1000 AND T/S 1500

by

Gregory C. Harder
P.O. BOX 6493
DENVER, COLORADO
80206

INTRODUCTION

This guide has been prepared in order to aid owners of T/S 1000 or T/S 1500 computers generate RLE graphics. RLE (*Run Length Encoded*) graphics is a method of transmitting encoded graphics information to your computer. Any computer capable of high resolution displays of at least 256 by 192 pixels can display RLE graphics, the T/S 1000 and T/S 1500 are no exceptions. All that is needed is the right combination of hardware/software and an RLE decoder program. This guide will describe a method of decoding the raw RLE data. If you follow the instructions you too can produce RLE graphics on your CRT terminal and printers.

HARDWARE REQUIREMENTS

The main hardware requirement for both the T/S 1000 and T/S 1500 is a suitable amount of RAM. This must include STATIC RAM in the 8K to 16K area, which is needed to produce the high resolution displays. This could be, for example, the "HUNTER BOARD" or "SCRAM BOARD". In addition at least 32K of user RAM must be available, i.e. 64K RAMPACKS.

If you intend to download your own RLE files from a remote terminal you will, of course, also need a modem.

RLE files are fairly memory intensive and can range from 3K to 28K or more in length depending on the picture complexity. This means, in the case of the T/S 1000 or 1500, that the largest whole file you can capture at one time is slightly less than 16K. In order to do this the entire 16K to 32K area must be open for the RLE download. The ideal scenario then is to run your terminal software below the 16K boundary or above the 32K boundary. However, assuming that the lower STATIC RAM will be reserved for high resolution displays then the terminal program must be placed above 32K.

In order to run machine code above 32K on the T/S 1000 you will need to install the M1 NOT modification, this is not required on the T/S 1500. If you do not install the M1 NOT modification then the maximum RLE file you can download will be on the order of 10K, the remaining RAM will be occupied by your terminal program.

SOFTWARE REQUIREMENTS

The method for decoding RLE graphics described herein is dependent upon the software used by the writer. Other programs may work, but I leave it up to you to make any changes in the decoder which this might necessitate.

The terminal program used is ZX-TERM-80. The main advantages to this program are the high resolution display modes, user friendliness, and relocatability. Relocate ZXT80 above 32K to open up all of the 16K for RLE data downloads.

The RLE decoder program needs a method for plotting the RLE picture. The RLE decoder therefore requires a copy of SRAM HIRES EXTENDED BASIC (SHREB). SHREB is an alternate operating system which permits full control of the high resolution

PROCEEDURE

The first thing we need to do is get an RLE file to decode. LOAD ZXT80 into your computer follow the start up proceedure and relocate to 32768. Find a BBS with some RLE files and download one. Hang up, the RLE file is now stored in the 0 REM DATA buffer, SAVE the file a couple of times to tape.

The RLE decoder needs to have the RLE data transfered to the address starting at 32768. We could add a BASIC PEEK/POKE program to do this, but this can be very slow, especially on larger RLE files. The first thing we'll do is add a short M.C. routine to do the transfer in a flash.

LOAD the 0 REM DATA buffer with the RLE file. Once the file is LOADed, POKE 16510,10. Now look at Listing 4, this is the M.C. we want to preface all the RLE files with. The 1 REM line must always be exactly 21 bytes long. Use the decimal dump to POKE the code into the REM line or, alternately, use the key board routine described in Listing 4. The key board method is most convenient especially if you will be doing alot of files. RAND USR 16514 will transfer the RLE file to the 32K area.

Once the RLE file is stored at 32K, LOAD SHREB HI-RES EXTENDED BASIC. Remove any extraneous lines and also delete line 2 as we will not need the 64 character print routines. Then POKE 18080,208 and 18081,75. Since SHREB is LOADed we could enter a decoder entirely in BASIC, if desired. Listing 1 is such a program, enter it then RUN to decode the RLE file already stored at 32K, however, be prepared to wait 10 to 30 minutes to see the final picture, in FAST mode.

A better method is to use the machine code RLE decoder shown in Listing 2. To enter this into SHREB delete all the BASIC decoder lines then create a 2 REM line of at least 131 bytes. Again, use a POKEr program to enter the decimal dump. Check your final product with the checksum routine shown. Finally, enter the lines from Listing 3 which is the BASIC portion of the M.C. decoder. SAVE to tape a couple of times then RUN. If everything was done correctly your RLE picture should be completed in under 30 seconds, in SLOW mode. You can even watch the picture being formed.

The M.C. decoder only returns to BASIC after the entire screen, 192 pixel rows, has been filled. Under normal conditions an RLE decoder should stop when a certain sequence of control codes is encountered at the end of the RLE file. This was not done on this M.C. decoder for a special reason.

As noted above, the maximum whole RLE file you can download is slightly less than 16K. However, files over 16K can be partially downloaded. If the RLE file is not too much over 16K then most of the picture will still be recovered. Obviously, files much larger than 16K may lose a significant portion of the picture and aren't worth downloading.

Since it is possible to download partial RLE files the ending control code sequence will be missing, this explains why the decoder does not test for them to locate the end of the file.

Some RLE files are meant to display only 256 by 176 resolution (T/S 2068 displays for example) since the ending control codes are not used by the decoder the bottom 16 lines of such a picture may not be correct. If it doesn't look right just use the SHREB scroll routines to erase the bottom part of the picture and then recenter it.

REFERENCES

Breunung, Gerd, 1986, 1000 One Chip Mod-A Built-in NVM: Syncware News, Vol.4,#1, p.18-21.

Article details construction of an internal STATIC RAM in the 8K to 16K area for the T/S 1000. Can be used for high resolution displays.

Fischer, Pete, and Ishi, Steve, 1987. The Guide to T/S Telecommunications.

Harder, Gregory, 1986, WRX16 Fix: Syncware News, Vol.4,#4, p.7.

Run WRX16 with 64K RAMPACKS.

Hopkins, Chris, 1986, Standard for RLE Files: TIME-I-CHANGE BBS download, User area DS:RLESTD.TXT

Leake, Stan, 1987, Run Length Encoded Graphics: Time Designs, Vol.3,#2, pp.17-20.

RLE decoder for the T/S 2068 computer.

Oliger, John, 1985, Run T/S 1000 Machine Code in High Ram: Syncware News, Vol.2,#5, p.10.

Article details how to install M1 NOT modification.

Rigter, Wilf, 1986, MRX16 HI-RES for the T/S 1000: Syncware News, Vol.4,#2, pp.13-18.

Describes high resolution operating system for the T/S 1000.

RESOURCES

Silicon Mountain Computers
C-12, Mtn. Station Group Box
Nelson, BC V1L 5P1
Canada

SCRAM BOARD- NVM, 8K to 16K, STATIC RAM Board for the T/S 1000 or T/S 1500. Can be used with high resolution software. Easily modified to allow for two switchable banks of 8K STATIC RAM or a RAM-EPROM combination.

ZX-TERM-80- High resolution terminal program for the T/S 1000-1500.

SRAM HI-RES EXTENDED BASIC- High resolution extended basic for the T/S1000-1500.

M1 NOT ADAPTOR- M1 NOT modification to run M.C. above 32K. No trace cutting or soldering, for T/S 1000 only.

Oodles of other quality high resolution and not so high resolution stuff for the T/S 1000-ZX81-T/S 1500.

Peter McMullin
2340 Queen St. E.
Toronto, Ontario M4E 1G9
Canada

Big-Printer SINCARTIST- Print those RLE screens to a big printer.

RLE FILES

TIME-X-CHANGE 213-329-3922 8/1/N 24hrs.

Compuserve

THE ZX-TERM EXCHANGE, c/o "Nicolson Nightime Network", (604) 354-4666 1800-0900 each night, all day Sunday. 8/N/1

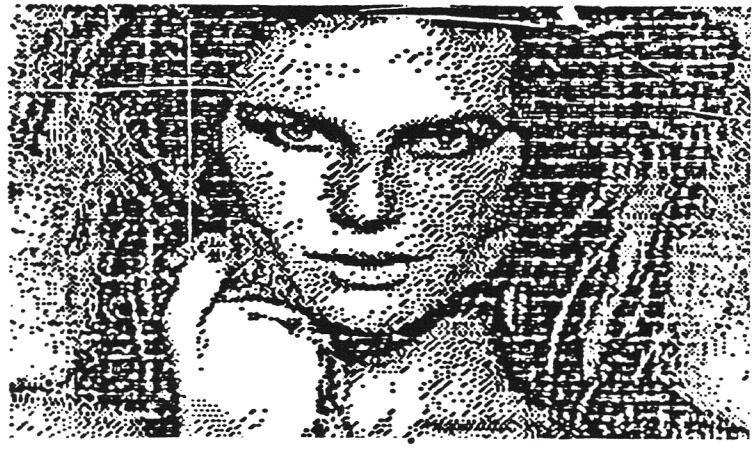
Listing 1: RLE DECODER BASIC LANGUAGE VERSION.

```
  0 REM [illegible]
  1 REM FAST
 10 REM RLE DECODER, MODIFIED
        FROM PROG. IN VOL. 3
        NO. 2 OF TIME DESIGNS
        MAGAZINE.
 20 REM RLE PICTURE DATA MUST
        START AT 32768, OR
        CHANGE VALUE AT LINE
        400.
 30 LET HR=19400
 40 IF USR HR THEN CLS
 50 IF USR HR THEN RUN
400 LET A=32768
410 LET X=0
420 LET Y=191
430 IF PEEK A<>71 THEN LET A=A+
1
440 IF PEEK A<>71 THEN GOTO 430
450 IF PEEK A<>72 THEN LET A=A+
1
460 IF PEEK A<>72 THEN GOTO 450
470 LET A=A+1
500 LET C=PEEK A-32
510 IF C<=0 THEN GOTO 560
520 LET X=X+C
530 IF X<=255 THEN GOTO 560
540 LET X=X-256
550 LET Y=Y-1
560 IF C<0 THEN STOP
570 LET A=A+1
580 LET C=PEEK A-32
590 IF C<0 THEN STOP
600 LET A=A+1
610 IF C=0 THEN GOTO 500
620 LET D=0
630 IF X<=255 THEN GOTO 660
640 LET X=X-256
650 LET Y=Y-1
660 IF USR HR THEN PLOT X,Y
670 LET D=D+1
680 LET X=X+1
690 IF D=C THEN GOTO 500
700 GOTO 630
```

Listing 2: RLE DECODER MACHINE CODE VERSION.

```
ADDR HEXCODE  NAME MNEMONIC
=========================
4E86~B7       REM2 OR A
4E87 B1            OR C
4E88 AA            XOR D
4E89 9B            SBC A,E
4E8A A9            XOR C
4E8B AA            XOR D
4E8C A8            XOR B
4E8D B4            OR H
4E8E A9            XOR C
4E8F AA            XOR D
4E90 B7            OR A
4E91 76            HALT
4E92 76            HALT
4E93 210080   DCOD LD HL,STOR
4E96 010000        LD BC,0000
4E99 1EBF          LD E,BF
4E9B 7E       LUP1 LD A,(HL)
4E9C FE47          CP 47
4E9E 2803          JR Z LUP2
4EA0 23            INC HL
4EA1 18F8          JR LUP1
4EA3 7E       LUP2 LD A,(HL)
4EA4 FE48          CP 48
4EA6 2803          JR Z INC>
4EA8 23            INC HL
4EA9 18F8          JR LUP2
4EAB 23       INC> INC HL
4EAC 7E       NEXT LD A,(HL)
4EAD D620          SUB 20
4EAF 281F          JR Z CONT
4EB1 381D          JR C CONT
4EB3 E5            PUSH HL
4EB4 2600          LD H,00
4EB6 6F            LD L,A
4EB7 09            ADD HL,BC
4EB8 44            LD B,H
4EB9 4D            LD C,L
4EBA 05            DEC B
4EBB 04            INC B
4EBC E1            POP HL
4EBD 2811          JR Z CONT
4EBF E5            PUSH HL
4EC0 60            LD H,B
4EC1 69            LD L,C
4EC2 010001        LD BC,0100
4EC5 A7            AND A
4EC6 ED42          SBC HL,BC
4EC8 44            LD B,H
4EC9 4D            LD C,L
4ECA 1D            DEC E
4ECB 3EFF          LD A,FF
4ECD BB            CP E
4ECE E1            POP HL
4ECF C8            RET Z
4ED0 23       CONT INC HL
```

```
ADDR HEXCODE  NAME MNEMONIC
=========================
4ED1 7E            LD A,(HL)
4ED2 D620          SUB 20
4ED4 23            INC HL
4ED5 28D5          JR Z NEXT
4ED7 1600          LD D,00
4ED9 04       BIT> INC B
4EDA 05            DEC B
4EDB 2813          JR Z SKP1
4EDD E5            PUSH HL
4EDE 60            LD H,B
4EDF 69            LD L,C
4EE0 010001        LD BC,0100
4EE3 A7            AND A
4EE4 ED42          SBC HL,BC
4EE6 44            LD B,H
4EE7 4D            LD C,L
4EE8 67            LD H,A
4EE9 3EFF          LD A,FF
4EEB 1D            DEC E
4EEC BB            CP E
4EED 7C            LD A,H
4EEE E1            POP HL
4EEF C8            RET Z
4EF0 F5       SKP1 PUSH AF
4EF1 E5            PUSH HL
4EF2 D5            PUSH DE
4EF3 C5            PUSH BC
4EF4 43            LD B,E
4EF5 1EFF          LD E,FF
4EF7 CD1041        CALL CK-Y
4EFA CDD141        CALL PLT?
4EFD C1            POP BC
4EFE D1            POP DE
4EFF E1            POP HL
4F00 F1            POP AF
4F01 14            INC D
4F02 03            INC BC
4F03 BA            CP D
4F04 28A6          JR Z NEXT
4F06 18D1          JR BIT>
```



ADDR	DECIMAL DATA					
20102	183	177	170	155	169	170
20108	168	180	169	170	183	118
20114	118	33	0	128	1	0
20120	0	30	191	126	254	71
20126	40	3	35	24	248	126
20132	254	72	40	3	35	24
20138	248	35	126	214	32	40
20144	31	55	29	229	38	0
20150	111	9	68	77	5	4
20156	225	40	17	229	96	105
20162	1	0	1	167	237	66
20168	68	77	29	62	255	187
20174	225	200	35	126	214	32
20180	35	40	213	22	0	4
20186	5	40	19	229	96	105
20192	1	0	1	167	237	66
20198	68	77	103	62	255	29
20204	187	124	225	200	245	229
20210	213	197	67	30	255	205
20216	16	65	205	209	65	193
20222	209	225	241	20	3	186
20228	40	166	24	209		

Fig. 1:DRAGON.RLE-T/S 2040

CHECKSUM PROGRAM

```
9500 LET X=0
9510 FOR N=20102 TO 20231
9520 LET X=X+PEEK N
9530 NEXT N
9540 IF X<>14041 THEN PRINT "CHE
CKSUM ERROR"
G000 REM


CHECKSUM=14041
```

Fig. 3:JAPANGRL.RLE-T/S 2040

Listing 3: BASIC PORTION OF M.C. RLE DECODER.

```
  0 REM [illegible]
  1 REM FAST
  2 REM [illegible]
 10 LET HR=19400
 20 IF USR HR THEN CLS
 30 IF USR HR THEN RUN
 40 IF USR 20115 THEN
    REM >RUN DECODER<
 50 IF USR HR THEN LPRINT I;
 60 IF INKEY$="" THEN GOTO 60
 70 IF USR HR THEN RETURN
 80 STOP
7999 REM **** SHREB SAVE ****
8000 IF USR HR THEN SAVE "RLE±DE
CODER",P
8010 GOTO 8120
8100 REM **** TIMEX SAVE ****
8110 SAVE "RLE.DECODER"
8120 LIST 20
9000 REM
```

```
      RLE DECODER
    G.C. HARDER 9/87
  STORE RLE DATA AT
    $8000 = 32768
  BEFORE RUNNING.
```

```
G000 REM
```

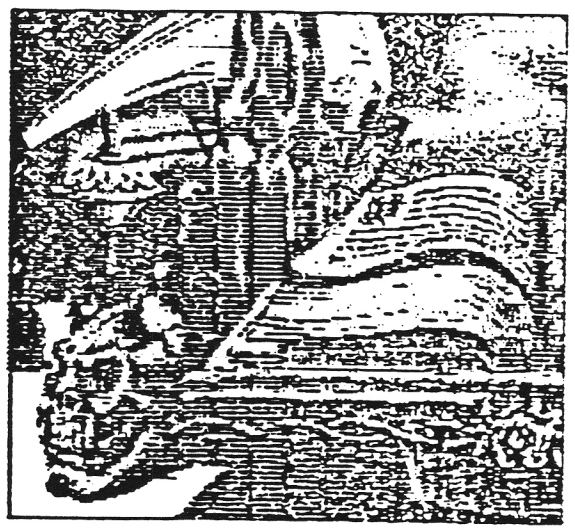


Fig. 2:MAGIC.RLE-T/S 2040

Listing 4: RLE DATA TRANSFER ROUTINE.

```
ADDR HEX CODE  NAME MNEMONIC
============================
4082 ED4B9A40 TRAN LD BC,(LEN)    ;Reg. BC=length of data REM
4086 0B            DEC BC         ;Adjust to exact length of
4087 0B            DEC BC         ;RLE data
4088 0B            DEC BC
4089 0B            DEC BC
408A 0B            DEC BC
408B 0B            DEC BC
408C 0B            DEC BC
408D 0B            DEC BC
408E 21A340        LD HL,RLE       ;Reg. HL=start addr. of RLE data
4091 110040        LD DE,DEST      ;Reg. DE=start addr. of storage
4094 EDB0          LDIR            ;Transfer RLE data to storage
4096 C9            RET             ;Return to basic
4097 76            HALT
4098 00            NOP
4099 0A            LD A,(BC)
409A 88        LEN ADC A,B
409B 2EEA          LD L,EA
409D 29            ADD HL,HL
409E 2639          LD H,39
```

BASIC part of routine.

```
   1 REM GOSUB ?■RND""""""""""5E■RN
D) ■ GOSUB ■TAN
  10 REM DATA
  20 RAND USR 16514
  30 STOP
  40 SAVE "DOLLAR.RLE"
```

Decimal Dump

Starting at address 16514.

```
237  75 154  64  11  11  11  11
 11  11  11  11  33 163  64  17
  0 128 237 176 201
```

Ending at address 16534.

This routine can be entered directly from the keyboard use the following key press sequence

KEY PRESS	RESULTS
1	1
E	REM
SHIFT 3	THEN
H	GOSUB
SHIFT 5	CURSOR LEFT
SHIFT 0	THEN DELETED
SHIFT 8	CURSOR RIGHT
SHIFT 9	G MODE
SPACE	INVERSE SPACE
SHIFT .	INVERSE COMMA
SHIFT ENTER	L MODE
SHIFT ENTER	F MODE
T	RND
SHIFT P	"
SHIFT P	"
SHIFT P	"
SHIFT P	"
SHIFT P	"
SHIFT P	"
SHIFT P	"
SHIFT P	"
5	5
SHIFT 9	G MODE
-	INVERSE 7
SHIFT ENTER	L MODE
SHIFT ENTER	F MODE
T	RND
SHIFT 0	)
SPACE	SPACE
SHIFT 9	G MODE
SPACE	INVERSE SPACE
SHIFT 9	L MODE
SHIFT 3	THEN
H	GOSUB
SHIFT 5	CURSOR LEFT
SHIFT 0	THEN DELETED
SHIFT 8	CURSOR RIGHT
SHIFT 9	G MODE
K	INVERSE K
SHIFT ENTER	L MODE
SHIFT ENTER	F MODE
E	TAN

Done, press ENTER to put line into listing then;

POKE 16515,75

Check to see that 1 REM looks exactly like the 1 REM shown in the listing above.

ENCODING A BINARY SCREEN FILE

Now that we know how to decode RLE files perhaps you would also like to be able to upload some of your own masterpieces, that you've created with SHREB, using the RLE standard. Again this is not too difficult. As with the decoder you will need a minimum of 64K of RAM.

PROCEEDURE

The first thing you have to do is have your graphic creation stored in the SHREB High Resolution Display File, from 8192 to 14335. Now look at Listing 5, this is the ENCODER. You will have to store this routine in your STATIC RAM board to use. You could use the decimal dump to POKE the values in each time, but this is prone to errors. A better method would be to use a BASIC program with some pseudo-"DATA" statements to hold the code and POKE it in automatically. I leave this chore up to you.

As shown, the ENCODER is located at 16128, you can put it at other locations if you want, it is free of CALLs and JUMPs. Do not store it somewhere in the HR-DFILE though!

What will happen when you CALL the routine by RAND USR 16128? The first thing it does is munch on your binary screen data and transform it into suitable RLE data. The RLE data is then stored at address 32768 and upwards. Once all the screen data is processed it will then automatically create a 0 REM DATA line of suitable size to hold the RLE file. If your RLE file is too large to store in 16K an error report 4 will result-"out of memory". If the file is not too large then it will be transported to the 0 REM DATA line. Once in the 0 REM DATA line the file is in a suitable form for uploading by ZXTBO as an ASCII file to any remote terminal. It can then be decoded by any RLE decoder program which tests for the proper RLE header, such as the one presented earlier.

As an example, once the RLE file is stored in the 0 REM DATA line SAVE it a few times to tape, then LOAD ZXTBO and relocate to 32768. Before going on line, reLOAD your RLE file. When it's LOADed re-enter ZXTBO, call up a BBS and do a normal upload. The upload will consist only of the encoded picture data. Also, your RLE file contains the proper ending code sequence.

As a note, before running the encoder you should NEW the computer to clear the entire BASIC area for the maximum possible RLE file.

LISTING 5: RLE ENCODER PROGRAM.

```
ADDR HEXCODE   NAME MNEMONIC
============================
3F00 CD230F    NCOD CALL FAST
3F03 FD3632[illegible] LD (FLGB),[illegible]
3F07 0E20           LD C,20
3F09 210080         LD HL,STOR
3F0C 110020         LD DE,HRDF
3F0F FD7136         LD (ON +),C
3F12 FD7137         LD (OFF+),C
3F15 3618           LD (HL),1B
3F17 23             INC HL
3F18 3647           LD (HL),47
3F1A 23             INC HL
3F1B 3648           LD (HL),48
3F1D 23             INC HL
3F1E FD363308 BYTS  LD (LOOP),08
3F22 EB             EX DE,HL
3F23 46             LD B,(HL)
3F24 23             INC HL
3F25 3E38           LD A,38
3F27 BC             CP H
3F28 EB             EX DE,HL
3F29 2857           JR Z END>
3F2B CB10      BITS RL B
3F2D 3031           JR NC BIT+
3F2F FDCB328E       RES 1,(FLGB)
3F33 FDCB3246       BIT 0,(FLGB)
3F37 2008           JR NZ SKP1
3F39 3A3640         LD A,(ON +)
3F3C 77             LD (HL),A
3F3D FD7136         LD (ON +),C
3F40 23             INC HL
3F41 FDCB32C6 SKP1  SET 0,(FLGB)
3F45 FD3437         INC (OFF+)
3F48 3E7E           LD A,7E
3F4A FDBE37         CP (OFF+)
3F4D 200A           JR NZ NXBT
3F4F 77        LUP1 LD (HL),A
3F50 23             INC HL
3F51 71             LD (HL),C
3F52 FD7136         LD (ON +),C
3F55 FD7137         LD (OFF+),C
3F58 23             INC HL
3F59 FD3533    NXBT DEC (LOOP)
3F5C 20CD           JR NZ BITS
3F5E 18BE           JR BYTS
3F60 FDCB3286  BIT+ RES 0,(FLGB)
3F64 FDCB324E       BIT 1,(FLGB)
3F68 2008           JR NZ SKP2
3F6A 3A3740         LD A,(OFF+)
3F6D 77             LD (HL),A
3F6E FD7137         LD (OFF+),C
3F71 23             INC HL
3F72 FDCB32CE SKP2  SET 1,(FLGB)
3F76 FD3436         INC (ON +)
3F79 3E7E           LD A,7E
3F7B FDBE36         CP (ON +)
3F7E 28CF           JR Z LUP1
3F80 18D7           JR NXBT
3F82 3618      END> LD (HL),1B
3F84 23             INC HL
3F85 3647           LD (HL),47
3F87 23             INC HL
3F88 364E           LD (HL),4E
3F8A 23             INC HL
3F8B 010080         LD BC,STOR
3F8E A7             AND A
3F8F ED42           SBC HL,BC
3F91 E5             PUSH HL
```

```
3F92 010C00          LD BC,000C
3F95 09              ADD HL,BC
3F96 44              LD B,H
3F97 4D              LD C,L
3F98 217C40          LD HL,407C
3F9B C5              PUSH BC
3F9C 03              INC BC
3F9D 03              INC BC
3F9E 03              INC BC
3F9F 03              INC BC
3FA0 CD9E09          CALL BCSP
3FA3 C1              POP BC
3FA4 23              INC HL
3FA5 23              INC HL
3FA6 3600            LD (HL),00
3FA8 23              INC HL
3FA9 3600            LD (HL),00
3FAB 23              INC HL
3FAC 71              LD (HL),C
3FAD 23              INC HL
3FAE 70              LD (HL),B
3FAF 23              INC HL
3FB0 36EA            LD (HL),EA
3FB2 23              INC HL
3FB3 3629            LD (HL),29
3FB5 23              INC HL
3FB6 3626            LD (HL),26
3FB8 23              INC HL
3FB9 3639            LD (HL),39
3FBB 23              INC HL
3FBC 3626            LD (HL),26
3FBE 23              INC HL
3FBF 3676            LD (HL),76
3FC1 23              INC HL
3FC2 3676            LD (HL),76
3FC4 23              INC HL
3FC5 C1              POP BC
3FC6 EB              EX DE,HL
3FC7 210080          LD HL,STOR
3FCA EDB0            LDIR
3FCC 1B              DEC DE
3FCD ED537640        LD (FEND),DE
3FD1 2A0C40          LD HL,(DFIL)
3FD4 2B              DEC HL
3FD5 3676            LD (HL),76
3FD7 C32B0F          JP SLOW
```

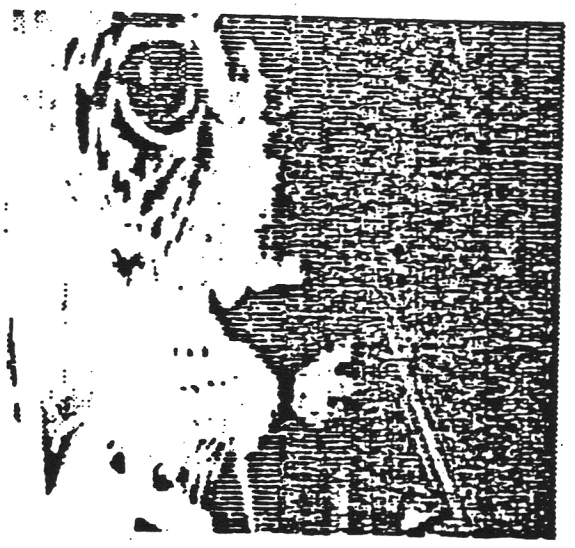


Fig. 4:LEOPARD.PLE-T/S 2040

ADDR	DECIMAL DATA					
16128	205	35	15	253	54	50
16134	0	14	32	33	0	128
16140	17	0	32	253	113	54
16146	253	113	55	54	27	35
16152	54	71	35	54	72	35
16158	253	54	51	6	235	70
16164	35	62	56	188	235	40
16170	87	203	16	48	49	253
16176	203	50	142	253	203	50
16182	70	32	8	58	54	64
16188	119	253	113	54	35	253
16194	203	50	198	253	52	55
16200	62	126	253	190	55	32
16206	10	119	35	113	253	113
16212	54	253	113	55	35	253
16218	53	51	32	205	24	190
16224	253	203	50	134	253	203
16230	50	78	32	8	58	55
16236	64	119	253	113	55	35
16242	253	203	50	206	253	52
16248	54	62	126	253	190	54
16254	40	207	24	215	54	27
16260	35	54	71	35	54	78
16266	35	1	0	128	167	237
16272	66	229	1	12	0	9
16278	68	77	33	124	64	197
16284	3	3	3	3	205	158
16290	9	193	35	35	54	0
16296	35	54	0	35	113	35
16302	112	35	54	234	35	54
16308	41	35	54	38	35	54
16314	57	35	54	38	35	54
16320	118	35	54	118	35	193
16326	235	33	0	128	237	176
16332	27	237	83	118	64	42
16338	12	64	43	54	118	195
16344	43	15				

CHECKSUM PROGRAM

```
9500 LET X=0
9510 FOR N=16128 TO 16345
9520 LET X=X+PEEK N
9530 NEXT N
9540 IF X<>20211 THEN PRINT "CHE
CKSUM ERROR"
```

CHECKSUM=20211

A Procedure For Downloading Crunched RLE Files To Any Computer System.

Most of the RLE picture files available from the GREYMATTER BBS have been crunched in order to conserve disk space. Normally, you would download the crunched file and then use an "uncrunch" utility to restore them. This is fine if you have a C/PM system, however if you do not then you must find an alternative method. Fortunately, there is a built-in "backdoor" method which can be used to automatically uncrunch the files. The procedure should be generally applicable to all computer systems, although the specifics will vary.

I have successfully used the method numerous times using a TIMEX/SINCLAIR 1500 computer and the terminal program ZX*TERM-80.

The key to uncrunching the RLE files is the C/PM "TYPE" command. The TYPE command is used to view/read the contents of a file. Some TYPEable files would be ones with the extensions such as TXT, DOC, BAS, HLP, etc. These, however, are not crunched files Extensions such as TZT, DZC, BZS, HZP, etc indicate that the files have been compressed.

For example, "README.TXT" and "README.TZT" would both be the same text file, but the latter has been compressed. Fortunately, the TYPE command can distingguish between TXT and TZT. The end result is that "TYPE README.TXT" and "TYPE README.TZT" will result in identical output.

Since RLE files are by design just ASCII encoded graphic information, if we could capture this data in uncrunched form the RLE picture could then be decoded in a normal fashion.

PROCEDURE

1. Select drive B5 which holds the RLE files.

2. Select a LBR file use "LUX RLE1" for example.

3. Initialize/clear/open your input capture buffer.

4. Select a crunched RLE file for capture, for example BBUNNY and TYPE it, your input line should look like;

   B5:RLE1.LBR-->TYPE BBUNNY.RZE

5. After pressing <cr> you must quickly disable the paging option by using CONTROL Z before the actual RLE data is sent, or the data will be corrupted by the paging prompts. This is probably the trickiest part of the method and may take some practice. The first three characters of the RLE data consist of ASCII codes 1B 47 48 (hex) which would appear on your terminal as <ESC> followed by GH. Thses are the opening header of the RLE file.

6. When the transfer is completed you'll see the system prompt again. At this point you should have an uncrunched copy of the RLE file stored in your input buffer. Save the file at this time for later decoding. follow the procedure again to capture another RLE file.

The most important items to remember are to clear/open your input capture buffer before TYPEing the file and shutting off the paging prompts.

Since "TYPE" is essentially a form of ASCII file transfer there is no built-in error checking on the data being sent. This is the biggest disadvantage of using this method for data transfer. This means you may get some errors in the RLE data, usually characterized by non-standard ASCII characters being displayed during the transfer, which can cause some drastic offsets in the final picture. These are repairable by using some graphic design programs. If you notice alot of line noise it is probably better to wait for some other time to transfer the RLE files.

Hope this helps.

Regards;

Gregory C. Harder Sept. 20, 1988